The **Computer Measurement Group**, commonly called **CMG**, is a not for profit, worldwide organization of data processing professionals committed to the measurement and management of computer systems. CMG members are primarily concerned with performance evaluation of existing systems to maximize performance (eg. response time, throughput, etc.) and with capacity management where planned enhancements to existing systems or the design of new systems are evaluated to find the necessary resources required to provide adequate performance at a reasonable cost.

This paper was originally published in the Proceedings of the Computer Measurement Group’s 2006 International Conference.

For more information on CMG please visit [http://www.cmg.org](http://www.cmg.org)
Use Trending to Manage Application and System Performance

Bruce Perkinson
ON-TEC Services

Bruce@ONTECSERVICES.COM

In the IT World we are all feeling the effects of some form of cost reduction. While companies may be successful in reducing staff costs they are also sacrificing legacy knowledge and technical expertise. This makes simplifying Performance Management more critical than ever. By collecting and trending a few key statistics, we can monitor the performance and then identify important trends for everything from batch cycle elapsed times to Netview CPU consumption. We'll review examples of how trending helped identify opportunities to reduce CPU consumption, shorten batch run times, and reduce cost of ownership.

This paper reviews the Performance Management process and introduces the use of trending graphs as a tool for performance analysis, problem diagnosis and reporting. Included in this paper is a brief discussion on the changing role of the Performance Analyst. From a technical perspective, we'll show how to use trending to make sense of the performance metrics being collected for mainframe applications and MVS system components.

Introduction

There are many sophisticated performance and capacity planning software packages available today. They provide extensive reporting and modeling capabilities but they are expensive and require skilled technical resources to manage. Not all companies can commit the money or the personnel required to benefit from these tools. Smaller installations have trouble justifying the expense.

Performance analysts, myself included, have spent thousands of hours diagnosing application and system performance problems. The urgency always seems sincere yet the variables always seem oddly familiar.

"Why are my test jobs running so long?"
"Job XXXX used to run 10 minutes and now it runs 40... we didn't change anything."
"Why did the on-lines come up 12 minutes late this morning?"
"When did Netview start using so much CPU?"
"What happened last month that caused my CPU bill to be so high?"

Answering these types of questions usually requires extracting weeks of historical performance statistics and a lot of painful data manipulation. Even if we’re fortunate enough to have historical performance databases we still have to spend the time and effort to isolate, extract, customize and analyze the data.

What the Performance analyst needs is a process to help them quickly and easily monitor key components of application and system performance. They need a process that requires less “hands-on” time but produces an end product that can easily be utilized, reviewed and communicated.

What’s Happened to Performance Management?

True performance management crosses the boundaries of applications and systems performance. A poorly performing system component, such as DB2, can be the cause of a poorly performing application. On the other hand, a poorly performing application can, in turn, be the major contributing factor in a poorly performing system.

Cost and management commitment have always hindered the effectiveness of performance management. In today’s world the major goal for an IT organization is to gain efficiencies and reduce costs. It’s perceived that lower cost application programmers paired with low cost systems programmers result in reducing overall processing costs. Unfortunately, junior programmers, overworked resources or lower cost skills usually translate into poorly performing applications and poorly tuned operating systems. The correct performance management process can ensure respectable performance.

What effects do changes in the “new world” have on the role of performance management today?

1 – COST. Companies demand lower priced application programmers and system programmers resulting in lower skill levels. Companies refuse to
pay the cost of extensive testing. Outsourcing companies bid contracts factoring low cost resources.

2 – More with less. IT budgets are shrinking. IT departments are shrinking. Rarely do we hear “is there anything I can do to improve the performance of my application?” or “I’ve been monitoring my application and I’ve allocated some project time to tune it after the implementation.” The performance analyst has to be self-motivated. They have to become the ‘Performance Police’. By this I mean they must monitor all aspects of application and systems performance.

3 – Who does the performance analyst work for? If the analyst works for the application teams how apt will they be to highlight costly performance problems? If the analyst works for tech support they don’t want to be involved in application issues. If the analyst works for the same company that is billing the customer for CPU consumption what incentive is there to reduce CPU consumption, especially if they are achieving their SLAs?

4 – Performance problems become as much a political issue as a technical one. Accepting ownership of a performance problem will cost the owner $$$$ that may not be budgeted. Remember, outsourcing business models succeed because they allocate a minimum number of hours for “run the business” and rarely do they allocate any hours for “tuning”.

**Add Trending Charts to the Performance Management Process**

Each company has its own unique IT requirements, especially when it comes to performance management. How you approach the process can translate into the familiar terms of PROACTIVE and REACTIVE. Some things just never go out of style.

Performance analysts will be more successful when they approach performance management with a proactive style. Being proactive really means determining in advance what performance metrics you’re going to capture and monitor. When a performance problem arises, you’ll know exactly where the related data resides and you’ll be able to extract it quickly and easily. Adding trend charts is a logical addition to proactive performance management. Trend charts will shorten the analysis process by providing a visual interpretation of data. Performance and consumption can quickly be interpreted as “trending up”, “trending down” or “flat”. Suspicious trends can be explored further.

The first step in the performance management process is to define the various types of performance data you need to collect. In most cases this will mean extracts from SMF or a SAS PDB. In other cases this may mean writing a custom program to collect specific performance data. The goal is to store the performance data in formats that can easily be trended.

The second step is to generate trend charts. The spreadsheet is the most common method used to create charts. You can import the data in a number of ways but developing macros would be the most efficient in the long term. Other tools such as SAS, MXG or shareware may be an alternative. Several software packages provide charting capabilities.

The third step is to monitor the trend charts on a regular basis. By monitoring I mean analyzing every chart looking for trends that merit further explanation. The actual frequency varies differently for each component you’re monitoring but the idea is to spot potential performance issues before they become performance problems. As a general guideline, use monthly trending for stable environments. Use a weekly trend for an unstable environment.

The final step to this simplified performance management process is to establish problem ownership. Identify who supports the root component and ensure everyone understands who the owner of the problem is. For example, if a production batch job is continually waiting for a DFHSM RECALL then the supporting application team should interface with the Storage Management team to eliminate the wait.

As a performance analyst you’ll be more productive if you don’t try to be the solution expert for every application and every component of system software. The focus is to manage performance.
How does this process work in the real world of performance management? Choose which of the following two scenarios you would prefer:

June 2nd
Dear Bob,

Just a quick note to mention that CPU utilization for the ABC application you support has been trending higher in the past several months. In the last 2 months there has been a 20% CPU increase. Also, please note job AR101P looks highly suspicious (see charts below).

If you have any questions please contact me!

Thanks,

Friendly proactive performance analyst

Or
August 12th

Dear Overworked Performance Analyst,

The customer was very unhappy to receive a 23% increase in CPU charges over previous months. This is the second consecutive month with this increase. I assured senior management that WE would be glad to review the data for the last 4 months and provide the customer with a detailed analysis explaining where the growth occurred… by C.O.B. this Friday.

![LPAR CPU Utilization Chart](chart)

Thanks in advance,

Noclue
Customer Relations

P.S. I’ll be out of the office the remainder of this week returning Monday

**What Data Should You Trend?**

You can trend almost anything... MVS CPU busy, CICS response times, CICS transaction dispatch wait times, DB2 Getpages, IMS calls, batch job statistics, TSO sessions, TCP/IP, FTP, ADABAS, SAR, VLF, DATACOM, ORACLE, ABENDAID, etc. The list is endless. And you aren’t limited to products that produce SMF records. All products have some sort of reporting capability. You can easily extract performance data from the appropriate report.

The type of data you decide to trend directly relates to your areas of responsibility. If you’re responsible for application performance then you’ll probably want to trend batch job START and STOP times, ELAPSED times, CPU times and maybe SERVICE UNITS. You may also consider adding statistics from DB2, IMS or whatever your DBMS is. If you’re responsible for CICS performance, you may want to trend response times, CPU times, I-Os and WAIT times. Again, you probably should consider DBMS statistics.

A seldom used but highly insightful source of statistical data for application monitoring is Epilog.

Epilog is a historical reporting component of OMEGAMON (which is now a member of the Tivoli family of products). For batch jobs I trend CPU Wait percent, CPU Used percent and the Productivity index. These statistics depict the health of a batch job. The trends show where there may be underlying performance issues.

The chart below tells us that the productivity of job XYZ has been decreasing over the last month. This indicates an increase in some type of wait.
A Productivity Index that is trending lower says that the percentage of time this job spends using the CPU is decreasing. It indicates that the job is spending a growing percentage of its run time outside of CPU processing. Further review is necessary to tell where this job is spending its time.

The two charts above help us see where job XYZ has been spending its time. The percentage of time waiting for CPU has been increasing. This means we have more research to do. Contributing factors could be that job XYZ needs a different WLM goal or the CPU is busier during this period than it was previously or I-O for this job has increased or jobs with a higher dispatching priority have increased or…

Choosing a Time Frame

The time frame you choose to trend will be partially determined by what you are monitoring. In some cases, such as monitoring the performance of a new application, you may want to trend on a daily basis. For quarterly or yearly reporting it may be best to trend on a monthly basis.

Keep the environment in mind when selecting trending time frames. System or application changes that occurred during a trending period can impact a trend. Trending too short of a time period or trending a timeframe that includes changes can skew the results.

Groupings – Creating a Logical View

One very useful trending technique is to combine multiple related entities into a logical view. Grouping can be done on an application level or at a system level. For instance, you could group all the CPU consumed by all the Accounts Receivable jobs into one logical view representing how much CPU the Account Receivable application consumed. Another example would be to create a Network group consisting of all the CPU consumed by TCPIPIP, Netview and Net. A high level view is perfect for communicating with management, working with technical support or capacity planning.

Below is an example of an application grouping for an Accounts Receivable application. Note how CPU utilization is trending higher.
Another interesting application trend is the combination of batch, online and database CPU consumption.

Grouping system tasks together can help uncover hidden trends. Note the increasing CPU trend for Netview in the next chart.

Grouping applications and system products into workloads can help explain where the CPU resources are being consumed.

Application Trending

There are many factors that impact application performance. The best way to monitor an application is to combine various application components into logical views and then trend the views. By looking at an application from its highest view we can determine if more detailed analysis is justified. If further investigation is unwarranted, we’ve saved ourselves
the time it would have taken to analyze down to the next level of granularity.

When trending applications we find that a lot of the common performance issues will identify themselves once we’ve charted a sufficient number of observations.

Let’s take the Inventory application as an example. First, let’s create a group called INVPROD. All Inventory batch jobs use “INV” for the first 3 characters of their jobname so we can group the CPU consumed for all the INV* jobs together. The name of the DB2 subsystem to support the Inventory application is DB2I. The names of the CICS regions supporting the Inventory application are CICSINVT and CICSINVA. The Inventory ADABAS task name is ADAINV.

If we want to show the CPU utilization trend for the Inventory application we trend all the CPU consumed by the entities that make up group INVPROD. This group depicts the combined CPU consumption for all the application and system components of the Inventory application.

In the next chart we are trending the elapsed time of the INV batch cycle.

Note the increase in application elapsed time for the batch processing directly corresponds to the increase in ADAINV CPU utilization in the prior chart. This application is doing more ADABAS work causing the batch jobs to elongate.

Also note the increase in system CPU busy relative to the increase in application elapsed time. We see we are driving the CPU to 100% busy. In addition to requiring more CPU to run, this application may be spending a lot of time waiting for available CPU resources. This application is also most likely impacting the throughput of all other batch cycles executing during this same time frame.

Remember, we can trend and compare anything….

System Trending

I always trend major operating system components. Since every performance problem is categorized as “a system problem” until proven otherwise, it’s prudent to have current charts reflecting system performance.
Grouping system components into one view is also an efficient way to monitor system maintenance and product upgrades. In the example below a company was positioning for a z/OS upgrade. They applied all the maintenance prerequisites over the weekend without incident. Below is the trend chart after 1 day.

The above chart is a condensed version of a 90-day daily chart. It’s important to understand your environment and the frequency of software changes. Time frames can be important. Note the Sunday and Monday CPU increase. Closer review shows that the started task SMSPDSE warrants further research. It has drastically increased in CPU consumption. Further investigation found that this was a known issue after applying maintenance and that changes to “SYS1.PARMLIB” member IGDMSxx would help reduce the impact until IBM provided a permanent fix.

Another “real life” example of how trending helped identify a potential headache. A shop upgraded Netview to a new release. Again, from a functionality standpoint, the upgrade was a success. By trending Netview’s daily CPU consumption it was identified that this release of Netview was consuming a higher percentage of CPU than the previous release. Further analysis found that several of the default parameters in the new release of Netview were different than the defaults settings in the previous release. This type of CPU increase may have gone unnoticed for several months resulting in increased costs that were avoidable.

When it comes to operating system performance, everyone expects the technical support team responsible for maintaining the operating system will also ensure that the operating system is running as efficiently as possible. In the application world the expectations for performance management are different. Rarely do we see a self-monitoring application team. This fact makes monitoring application performance a critical component of performance management.

A majority of the performance issues I’ve experienced originate in the application arena. The leading violators are usually excessive I-O followed by high CPU consumption. Of course, high I-O usually drives higher CPU usage.

Protect your technical integrity. Here’s an excellent justification for actively trending system components. The final step in diagnosing the “ownership” of any performance problem should be to understand what, if any, impact “system” components have on a performance problem. For example, when diagnosing a problem where a user is complaining of slow response time in a CICS/DB2 transaction, make sure you investigate all possible suspects. You may find that the trend charts for this transaction shows a large increase in DB2 calls and CPU consumption over the last 2 months. But, further “system” investigation also finds that DB2 THREAD WAITS in this CICS are trending higher due to the fact that transactions are issuing more DB2 calls and occupying the threads longer. The root cause of this problem is the increase in DB2 calls but the thread waits are definitely a contributing factor. When discussing your diagnosis
with the owner of this suspect transaction, be sure to explain that the DB2 support team will take ownership of the DB2 thread issue!

Management Reporting

Graphical reporting is perfect for management reporting. Monthly and quarterly graphs can be used to provide a graphical representation of resource utilization. Management can see where processing costs are rising. Similar to a Red / Yellow / Green report or a dashboard, these graphs give management a quick, high-level overview.

Real Life Story – Trending at its Best

This is a true story. A company requested an outside “opinion” after they had already added 4 additional processors in a 6-month time period.

A new WEB/CICS/DB2 application, let’s call it application XYZ, was designed and implemented with little regard for performance. To defray costs an “influential” user enlisted an outsourcing firm to develop a new critical business application. They designed and tested the code in a vacuum with little input from the legacy IT staff. The application was ready for production. No load testing, no performance measurements, no technical review.

East coast users were migrated to the new application in a “phased” approach. The philosophy being they will “test” in production and can always move the users back to the old application if necessary. The first group of users produced a small application load and had a minor impact to the overall system. Added the second group of users. CICS is a little slow… out came the email… “A SUCCESS at 40% of the COST of an in-house developed application”. The next groups of users were added. The Oracle server broke. Installed a bigger Oracle server. “It’s a system problem, CICS isn’t getting enough CPU”. Political frenzy. “We were planning for a CPU upgrade next fiscal year anyway… add an engine”. Add the west coast users. The Oracle server broke. Installed a bigger Oracle server. “CICS is a little slow”. “The nightly batch cycle isn’t completing on time, the onlines are coming up late… maybe we should back off the west coast users so we can bring the application down earlier”. Political frenzy. “Add an engine”. Another new CICS/DB2 application, YYY, is implemented. “It’s a system problem, CICS isn’t getting enough CPU”. “We had planned a CPU upgrade for the YYY application next quarter anyway, add an engine”. Life is good, although getting more expensive. The remaining users were added to the XYZ application. Add the midwest region. The Oracle server (now with the power of a mainframe) is holding up. “CICS response time is unacceptable from 10 am to 4 pm when both east coast and west coast are using the application at the same time”. Political finger pointing begins. “It’s a system problem, CICS isn’t getting enough CPU… bring in an outside consultant to tune the system… and add an engine”. The 4th engine was added and the performance is “acceptable”. Software licensing costs are higher than ever expected.

Thankfully I had developed the trending methodology prior to this company contacting me. Diagnosing this problem was going to be simple. Due to the political nature of the application’s origin and the money spent on hardware and software the presentation of the results would definitely require tact.

Watching real-time performance monitors gave quick insight as to the behavior of the CICS transactions, the amount of CPU the transactions consumed and the volume of DB2 calls they executed. We pulled CPU statistics, CICS statistics and DB2 statistics for the past 9 months. This gave us 3 months of data prior to the implementation of the XYZ application. It was fortunate that this company had daily summary statistics available.

Now let’s review some of the trending charts that were produced to diagnose the perceived “system problem”.

CPU utilization trend charts show that there was a direct correlation between the overall increase in CPU utilization and the implementation of the new application. We also can see that as additional users were added to the application, correspondingly, CPU consumption increased and the performance issues accelerated.

![LPAR1 CPU Utilization 8am to 5pm](image-url)
Note the CPU increases in “CICS prod” and “prod batch” beginning in May.

The weekly CICS trending charts show that performance problems were imminent immediately following the implementation of application XYZ into production. The chart below clearly shows a significant increase in CPU consumption.

CPU utilization increased in May and grew as additional users were added to the application.

As the number of users grew so did the count for the new “WXYZ” transaction.

The next three charts show the impact that transaction “WXYZ” had on resource consumption. Notice the correlation between transaction WXYZ CPU consumption, CPU consumption by DB2 plan WXYZ and DB2 GETPAGE activity by DB2 plan WXYZ.

Together these trend charts show us that there has been a dramatic increase in CPU utilization that began in May and continued to increase each month. Though the increased CPU utilization shows up in CICS, the DB2 GETPAGE trend tells us that the CPU increase is actually being driven by the increase in DB2 activity by transaction WXYZ (the DB2 CPU being used by a CICS transaction is charged to CICS). To support our diagnosis we can see that as the CICS transaction count decreases in July, so does the DB2 GETPAGE count and CICS CPU utilization.
To improve the efficiency of this application several new DB2 indices were added and critical CICS transactions were redesigned. As a result the DB2 I/Os were reduced dramatically and response times approached sub-second.

Users were eventually convinced that this was not a system performance problem but a classic case of a poorly designed application suffering from excessive I-O.

What is the lesson learned here (ignoring the poor choice of eliminating performance testing from the project plan)? The lesson is that this company could have avoided several CPU upgrades and incremental software costs had they used simple performance management techniques.

Trending is now a very active part of this company’s performance management philosophy.

**Summary and Conclusion**

Since the world of the mainframe is still evolving, the need for efficient performance management is more important than ever! IT departments are continually changing and workloads are increasing. Skill levels may not be what they used to be. Companies have multiple vendor solutions that require more precise diagnosis. Performance analysis may only be on a quarterly basis or even provided by an independent entity.

The conclusion is simple. Proactive performance management is essential. This article illustrated how trending performance statistics identified performance issues before they became performance problems.

By including trending in the performance management process we become more proactive and more efficient. Trending will “visually” highlight performance issues as well as provide the statistical data required to communicate a problem to its owner.

Trending graphs provide an easy to understand graphical picture. Detailed graphs can be used to highlight application growth to application programmers. Summary graphs are useful in high-level communication with senior management.

Trending works.